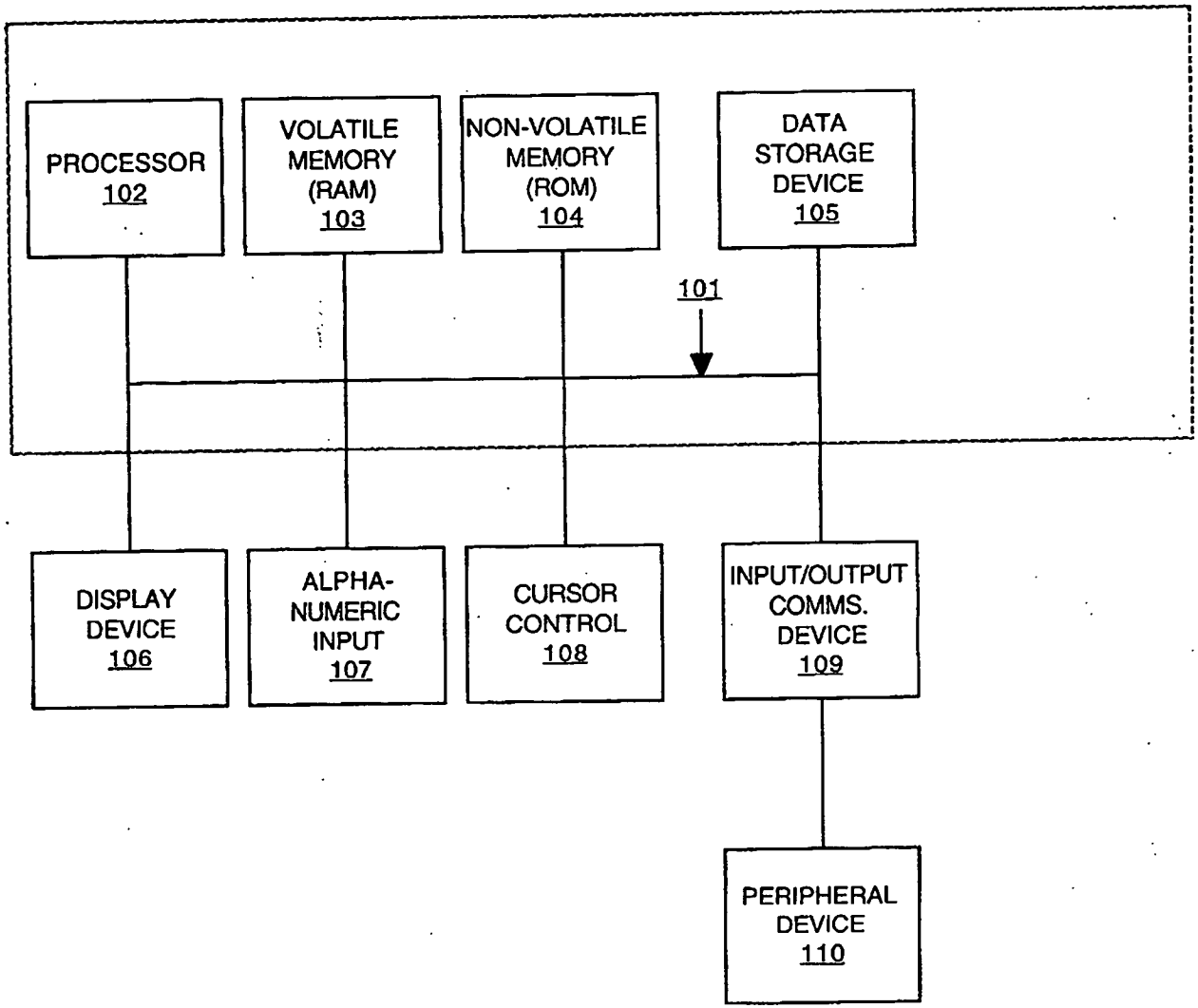
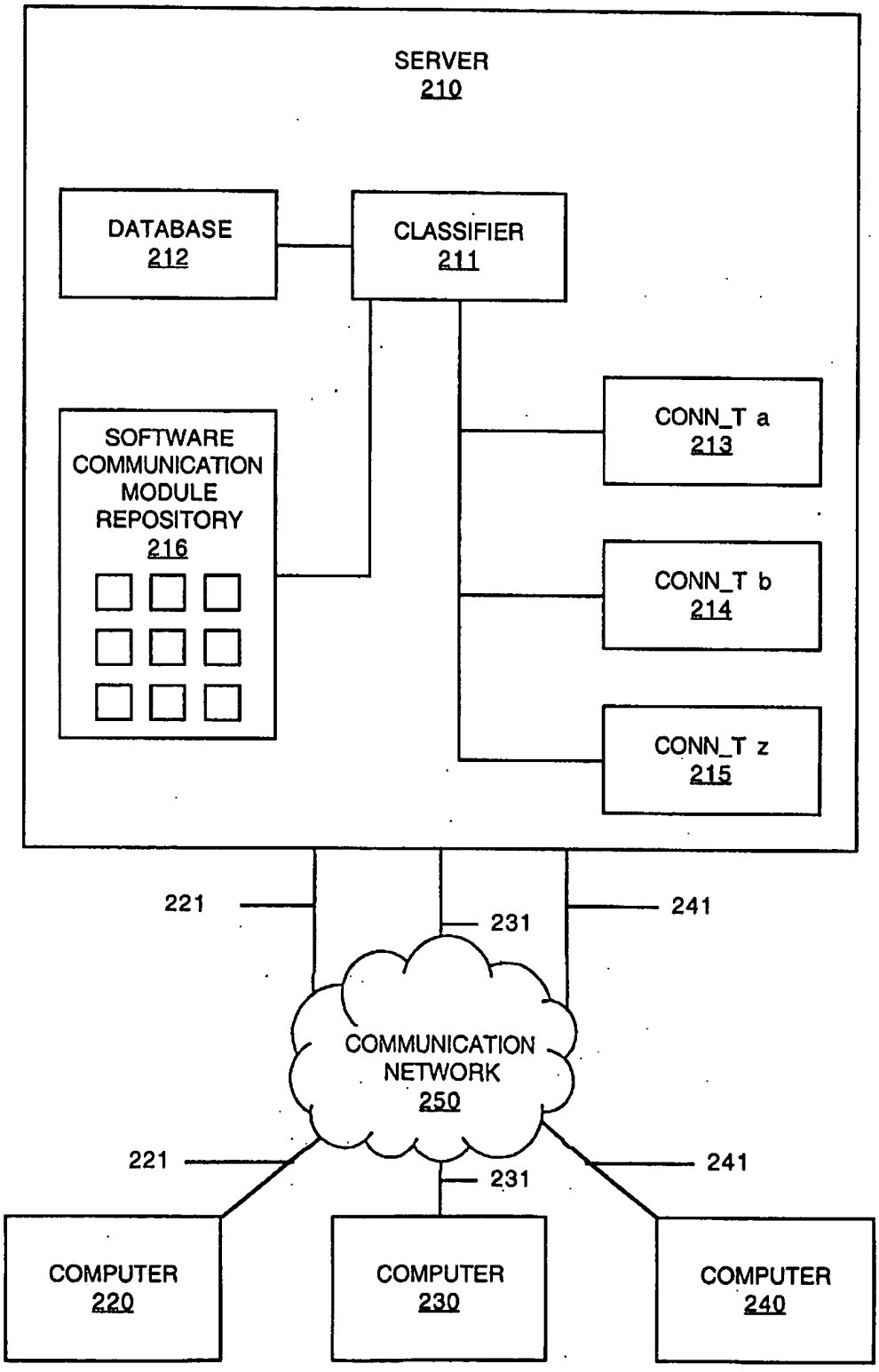
100

FIGURE 1


PROCESSOR
102
VOLATILE MEMORY (RAM)
103
NON-VOLATILE MEMORY (ROM)
104
DATA STORAGE DEVICE
105
101
DISPLAY DEVICE
106
ALPHA-NUMERIC INPUT
107
CURSOR CONTROL
108
INPUT/OUTPUT COMMS. DEVICE
109
PERIPHERAL DEVICE
110

200

FIGURE 2


SERVER
210
DATABASE
212
CLASSIFIER
211
SOFTWARE
COMMUNICATION
MODULE
REPOSITORY
216
CONN_T a
213
CONN_T b
214
CONN_T z
215
221
231
241
COMMUNICATION
NETWORK
250
221
231
241
COMPUTER
220
COMPUTER
230
COMPUTER
240

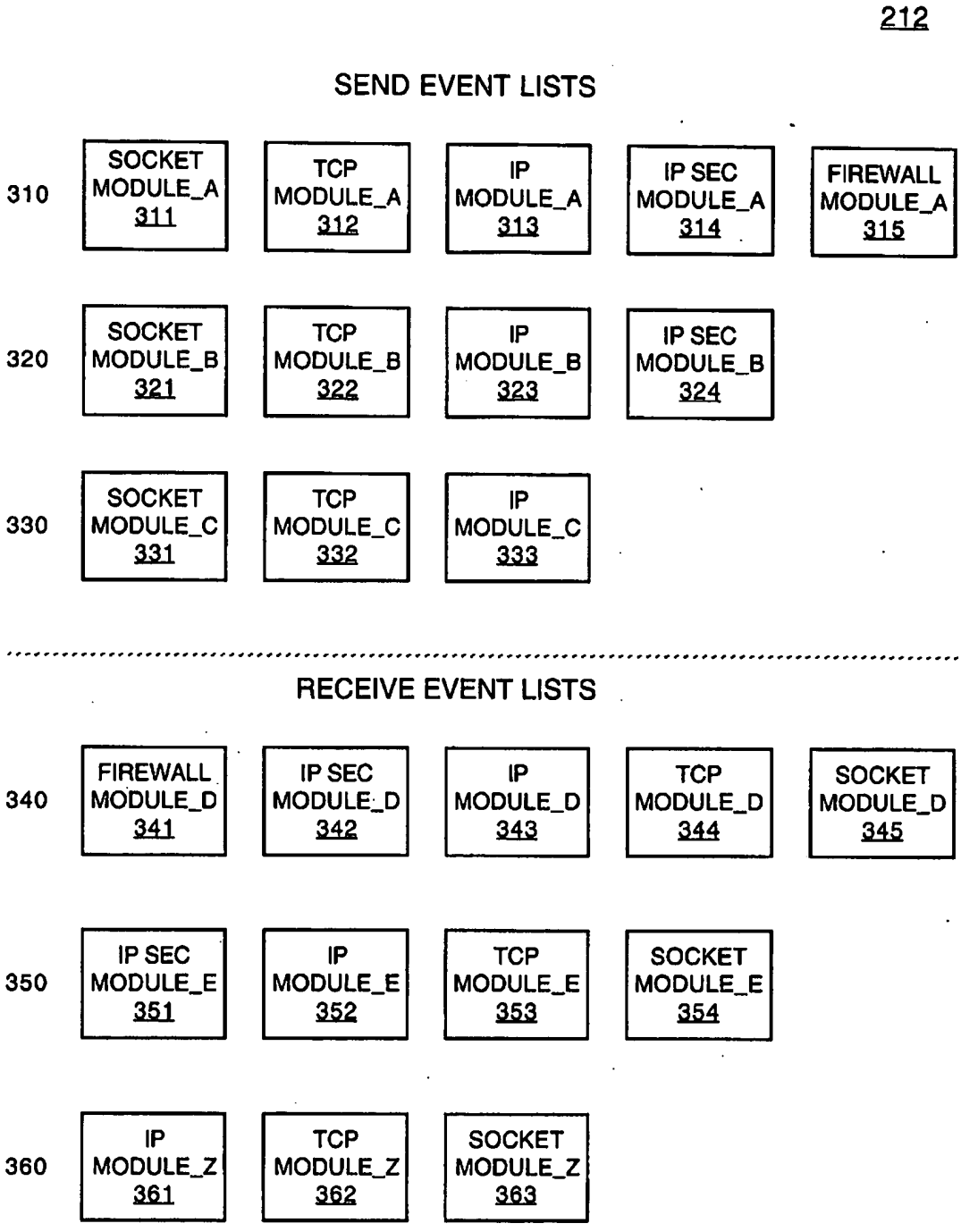

212
SEND EVENT LISTS
310
SOCKET MODULE_A 311
TCP MODULE_A 312
IP MODULE_A 313
IP SEC MODULE_A 314
FIREWALL MODULE_A 315
320
SOCKET MODULE_B 321
TCP MODULE_B 322
IP MODULE_B 323
IP SEC MODULE_B 324
330
SOCKET MODULE_C 331
TCP MODULE_C 332
IP MODULE_C 333
RECEIVE EVENT LISTS
340
FIREWALL MODULE_D 341
IP SEC MODULE_D 342
IP MODULE_D 343
TCP MODULE_D 344
SOCKET MODULE_D 345
350
IP SEC MODULE_E 351
IP MODULE_E 352
TCP MODULE_E 353
SOCKET MODULE_E 354
360
IP MODULE_Z 361
TCP MODULE_Z 362
SOCKET MODULE_Z 363

Fig. 3A

360

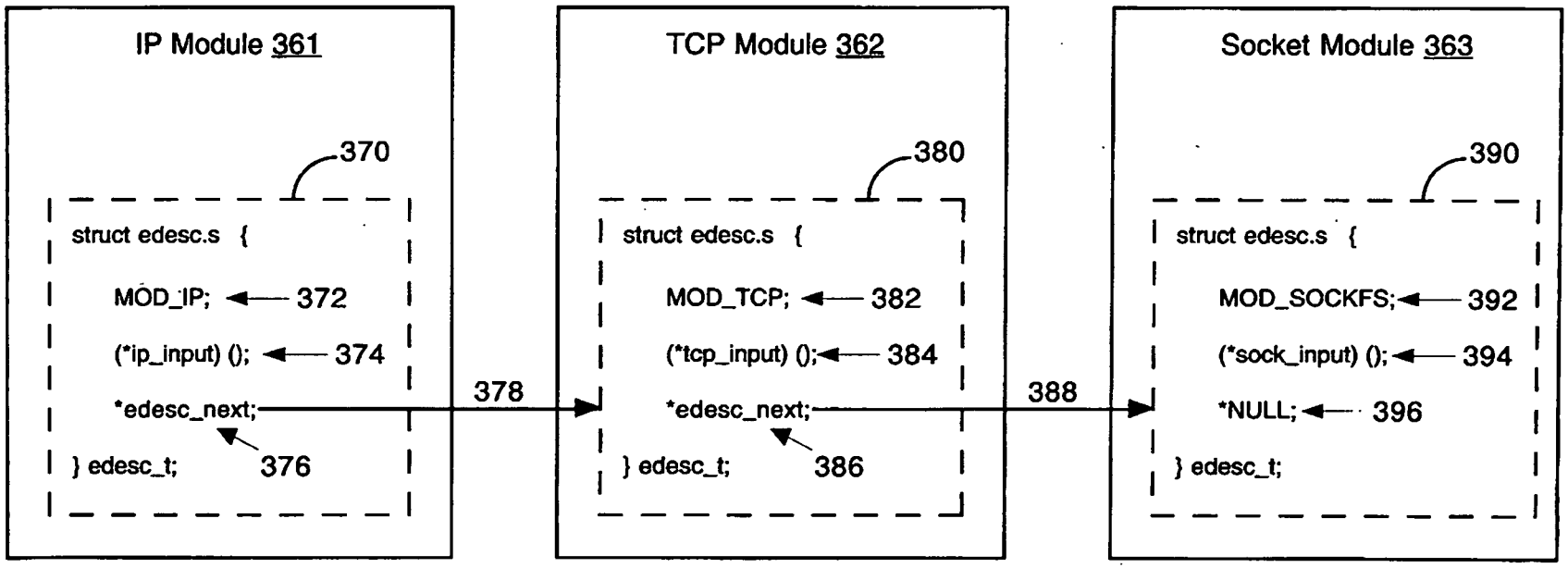

IP Module 361
370
struct edesc.s {
MOD_IP; 372
(*ip_input) (); 374
*edesc_next; 376
} edesc_t;
378
TCP Module 362
380
struct edesc.s {
MOD_TCP; 382
(*tcp_input) (); 384
*edesc_next; 386
} edesc_t;
388
Socket Module 363
390
struct edesc.s {
MOD_SOCKFS; 392
(*sock_input) (); 394
*NULL; 396
} edesc_t;

Fig. 3B

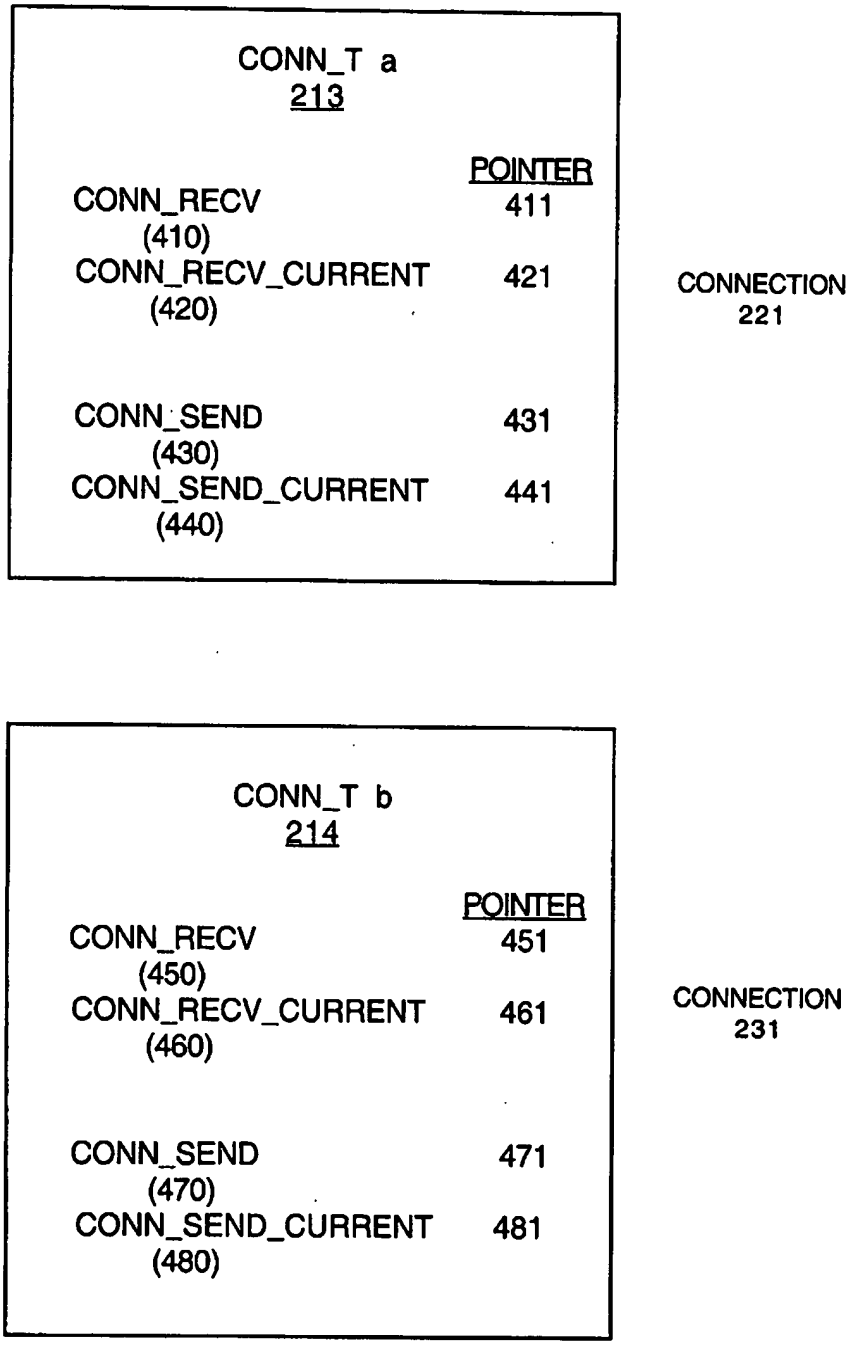

CONN_T a
213
POINTER
CONN_RECV
(410)
411
CONN_RECV_CURRENT
(420)
421
CONNECTION
221
CONN_SEND
(430)
431
CONN_SEND_CURRENT
(440)
441
CONN_T b
214
POINTER
CONN_RECV
(450)
451
CONN_RECV_CURRENT
(460)
461
CONNECTION
231
CONN_SEND
(470)
471
CONN_SEND_CURRENT
(480)
481

Fig. 4

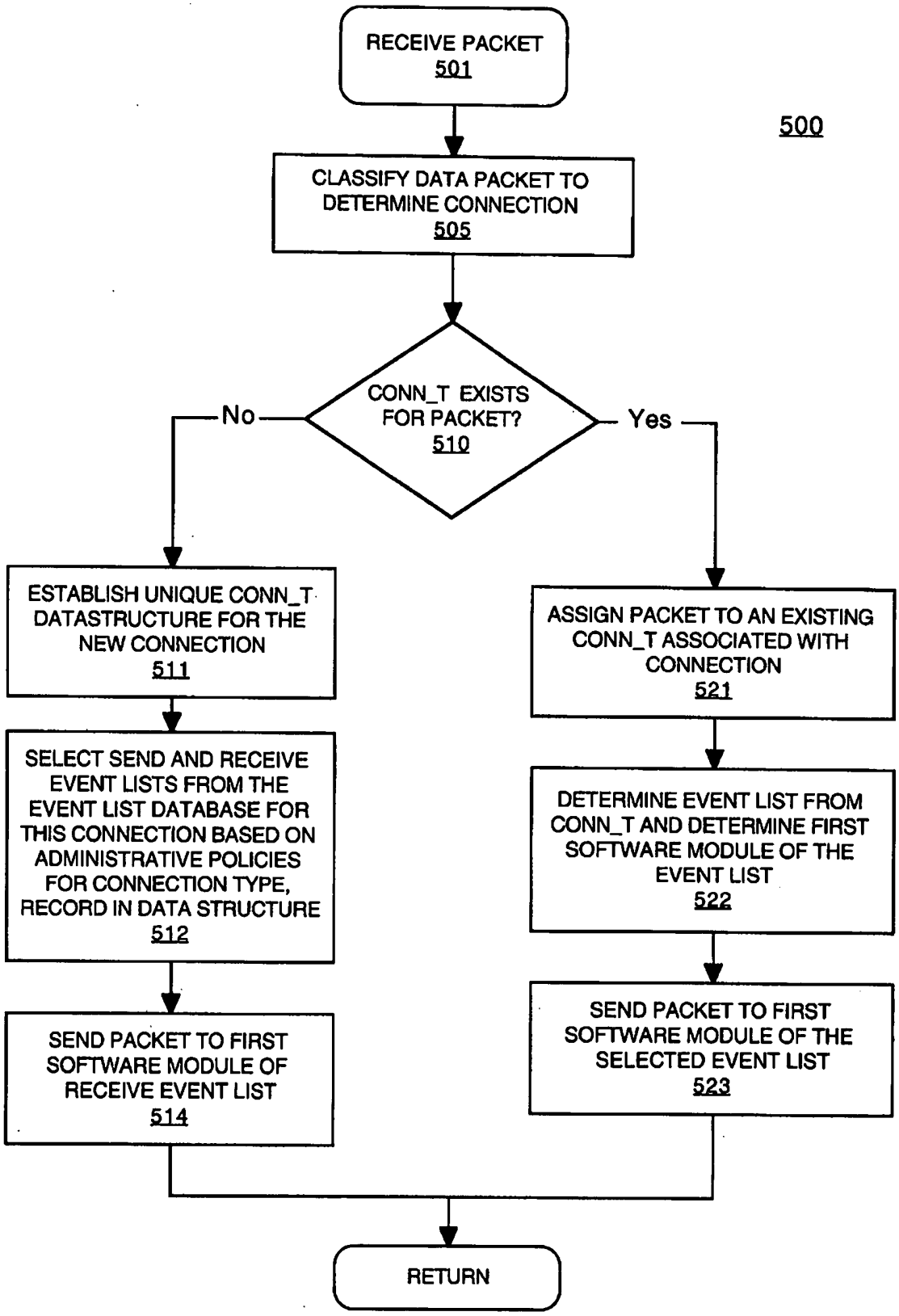

RECEIVE PACKET
501
500
CLASSIFY DATA PACKET TO DETERMINE CONNECTION
505
CONN_T EXISTS FOR PACKET?
510
No
Yes
ESTABLISH UNIQUE CONN_T DATASTRUCTURE FOR THE NEW CONNECTION
511
SELECT SEND AND RECEIVE EVENT LISTS FROM THE EVENT LIST DATABASE FOR THIS CONNECTION BASED ON ADMINISTRATIVE POLICIES FOR CONNECTION TYPE, RECORD IN DATA STRUCTURE
512
SEND PACKET TO FIRST SOFTWARE MODULE OF RECEIVE EVENT LIST
514
ASSIGN PACKET TO AN EXISTING CONN_T ASSOCIATED WITH CONNECTION
521
DETERMINE EVENT LIST FROM CONN_T AND DETERMINE FIRST SOFTWARE MODULE OF THE EVENT LIST
522
SEND PACKET TO FIRST SOFTWARE MODULE OF THE SELECTED EVENT LIST
523
RETURN

Fig. 5A

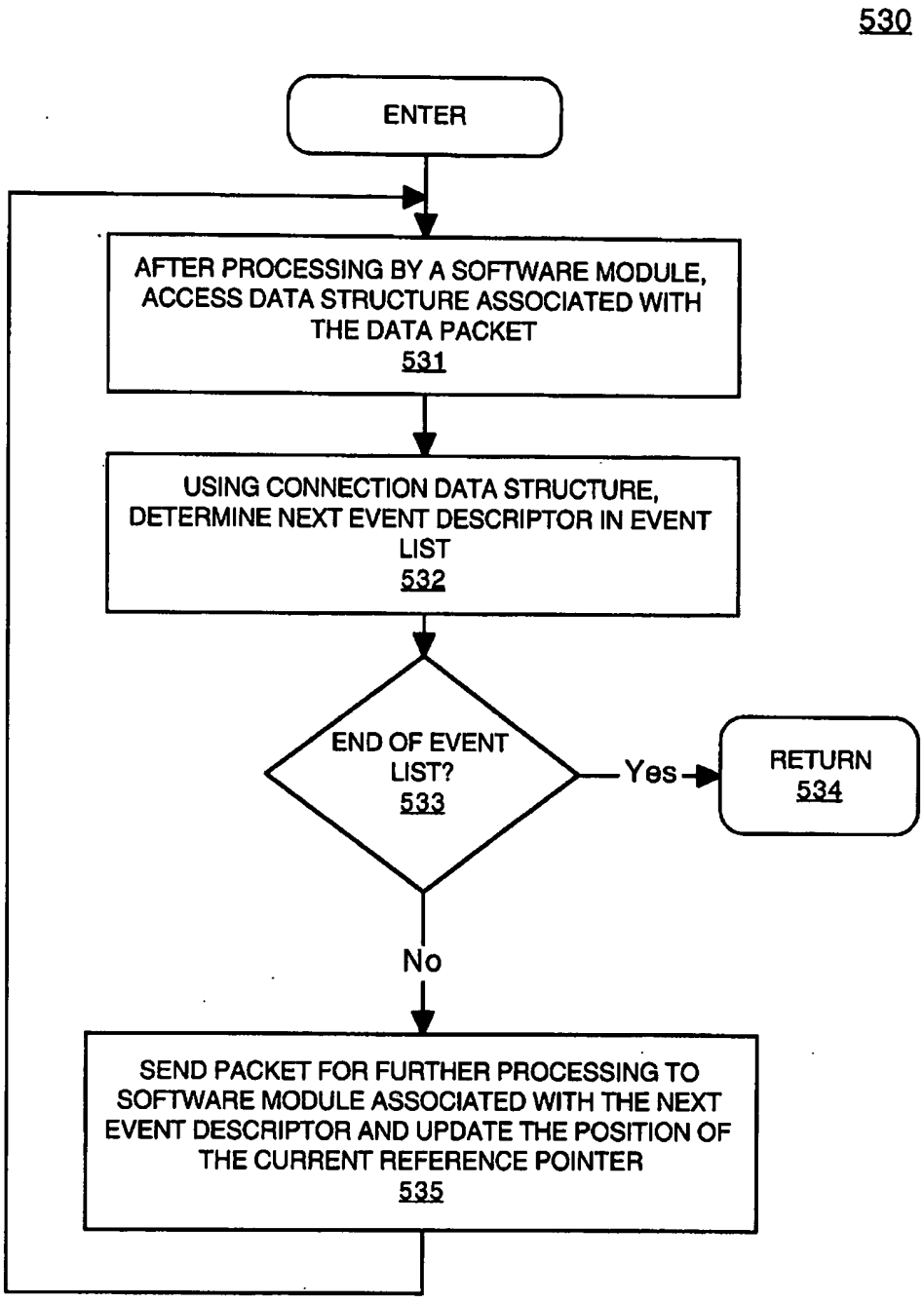
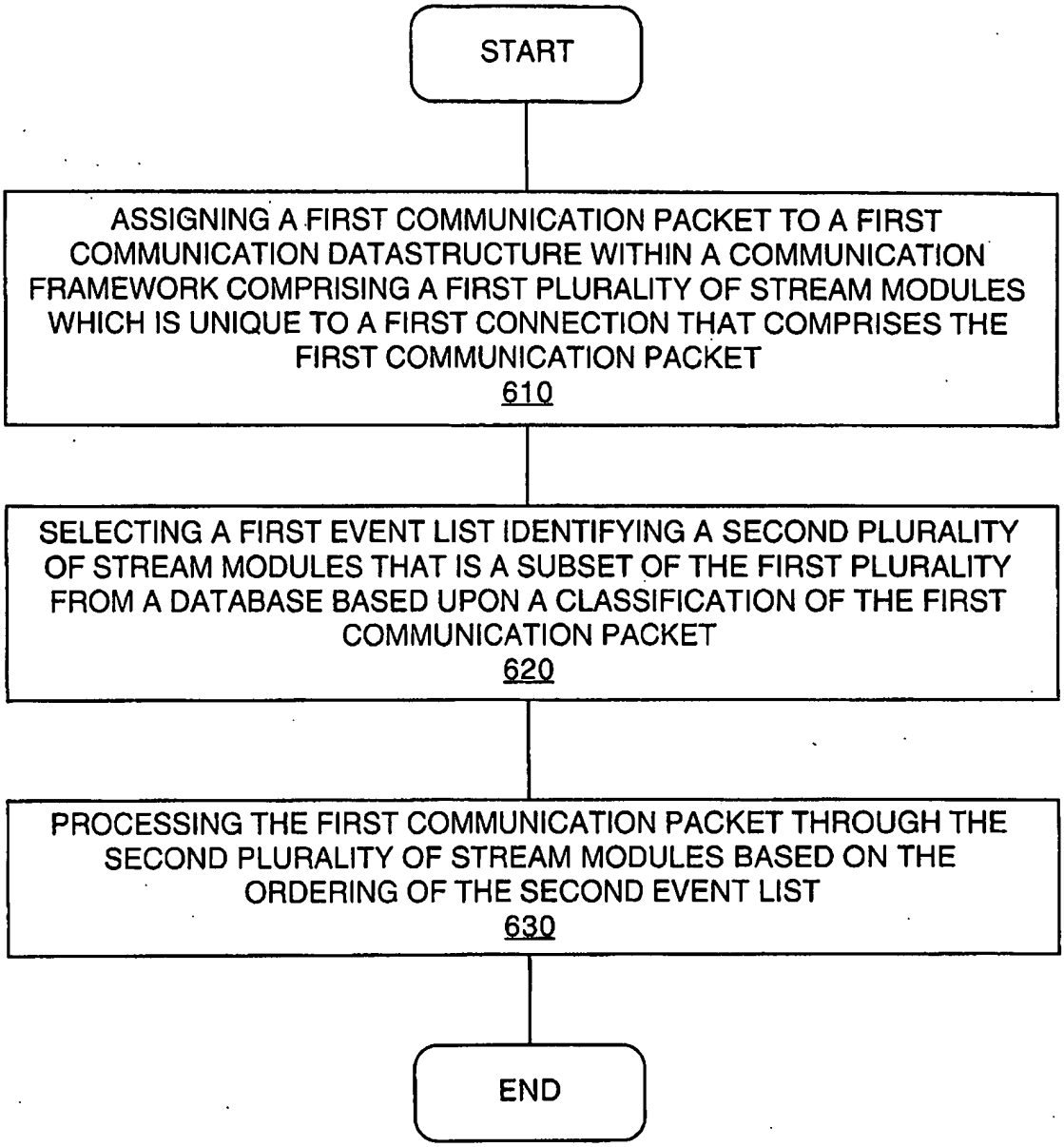

530
ENTER
AFTER PROCESSING BY A SOFTWARE MODULE, ACCESS DATA STRUCTURE ASSOCIATED WITH THE DATA PACKET
531
USING CONNECTION DATA STRUCTURE, DETERMINE NEXT EVENT DESCRIPTOR IN EVENT LIST
532
END OF EVENT LIST?
533
Yes
RETURN
534
No
SEND PACKET FOR FURTHER PROCESSING TO SOFTWARE MODULE ASSOCIATED WITH THE NEXT EVENT DESCRIPTOR AND UPDATE THE POSITION OF THE CURRENT REFERENCE POINTER
535

Fig. 5B

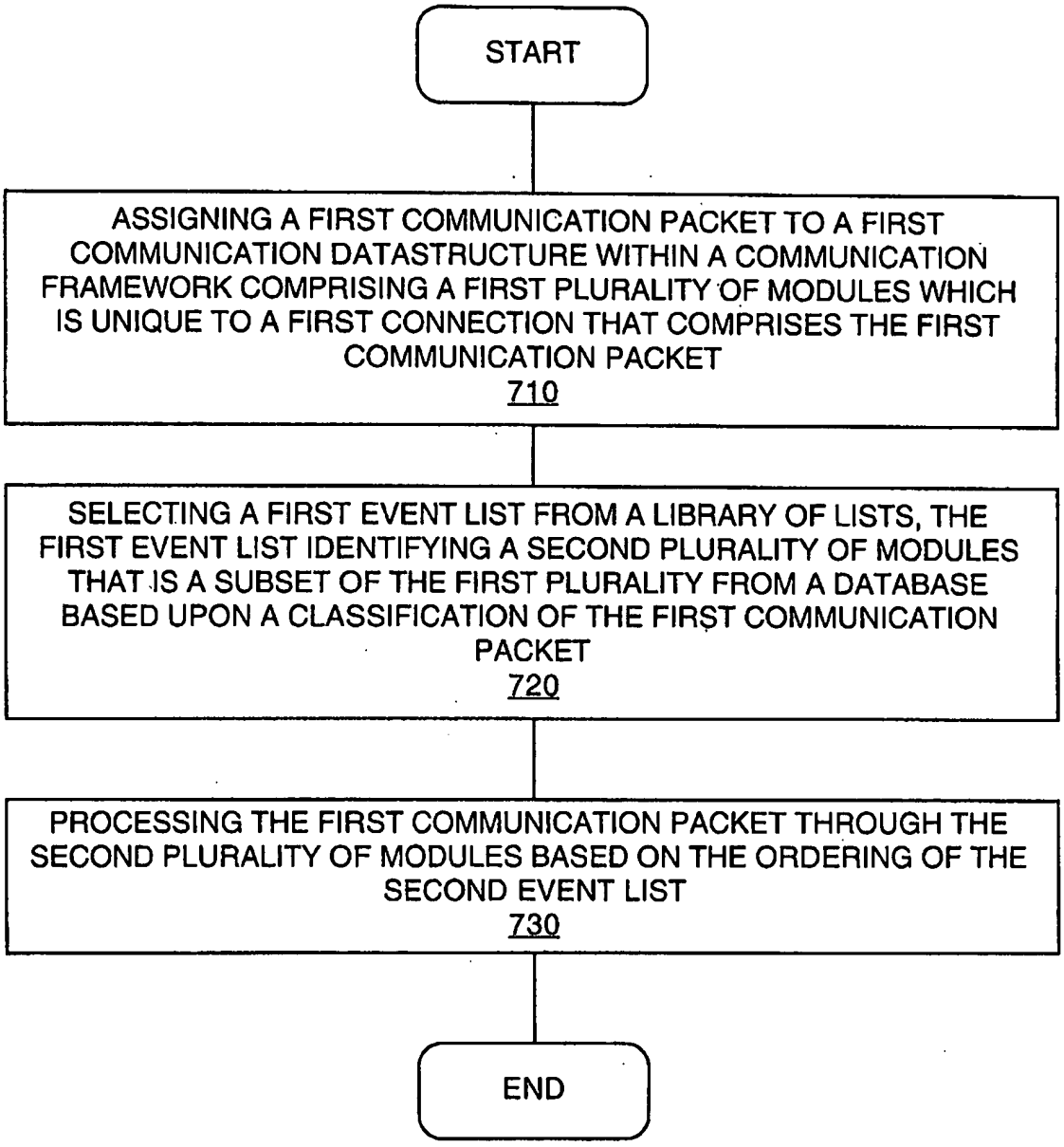
600

FIGURE 6


START
ASSIGNING A FIRST COMMUNICATION PACKET TO A FIRST COMMUNICATION DATASTRUCTURE WITHIN A COMMUNICATION FRAMEWORK COMPRISING A FIRST PLURALITY OF STREAM MODULES WHICH IS UNIQUE TO A FIRST CONNECTION THAT COMPRISES THE FIRST COMMUNICATION PACKET
610
SELECTING A FIRST EVENT LIST IDENTIFYING A SECOND PLURALITY OF STREAM MODULES THAT IS A SUBSET OF THE FIRST PLURALITY FROM A DATABASE BASED UPON A CLASSIFICATION OF THE FIRST COMMUNICATION PACKET
620
PROCESSING THE FIRST COMMUNICATION PACKET THROUGH THE SECOND PLURALITY OF STREAM MODULES BASED ON THE ORDERING OF THE SECOND EVENT LIST
630
END

700

FIGURE 7


START
ASSIGNING A FIRST COMMUNICATION PACKET TO A FIRST COMMUNICATION DATASTRUCTURE WITHIN A COMMUNICATION FRAMEWORK COMPRISING A FIRST PLURALITY OF MODULES WHICH IS UNIQUE TO A FIRST CONNECTION THAT COMPRISES THE FIRST COMMUNICATION PACKET
710
SELECTING A FIRST EVENT LIST FROM A LIBRARY OF LISTS, THE FIRST EVENT LIST IDENTIFYING A SECOND PLURALITY OF MODULES THAT IS A SUBSET OF THE FIRST PLURALITY FROM A DATABASE BASED UPON A CLASSIFICATION OF THE FIRST COMMUNICATION PACKET
720
PROCESSING THE FIRST COMMUNICATION PACKET THROUGH THE SECOND PLURALITY OF MODULES BASED ON THE ORDERING OF THE SECOND EVENT LIST
730
END